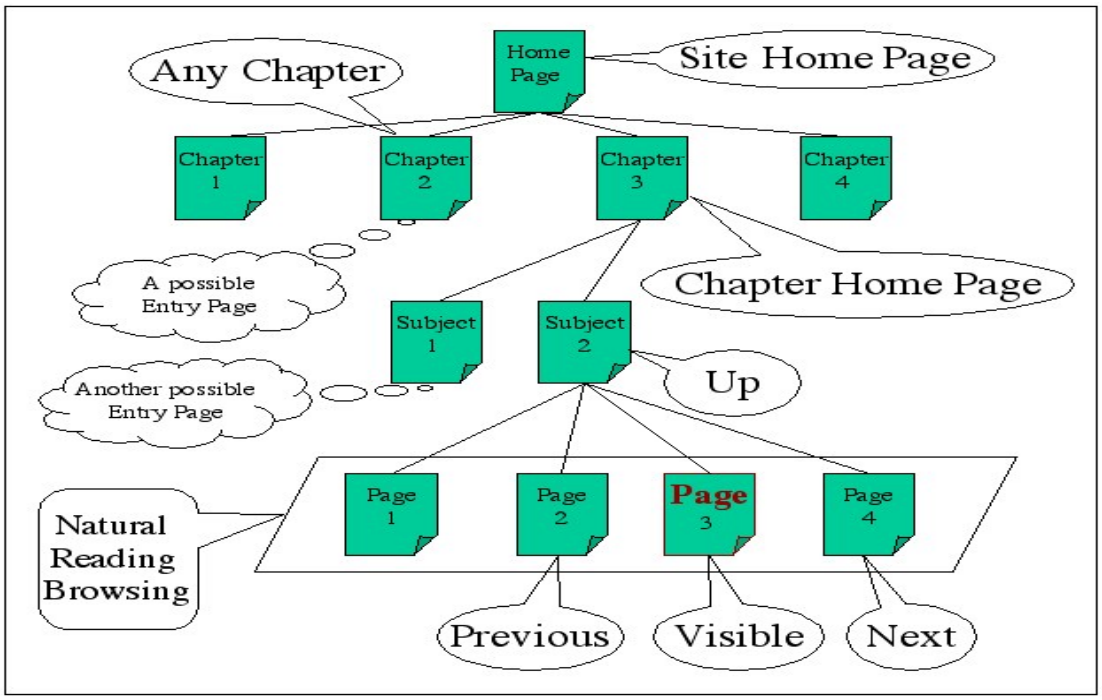
COMP6701 eScience Project

Organising A Navigation Hierarchy Using The WebTree Application.



Student Name - Prabhath Chinthaka Waduge.
Student ID - U3928468.
Supervisor - Mr. Pascal Vuylsteker.
Course Convenor - Dr. Henry Gardner.

TABLE OF CONTENTS Page No

Acknowledgement 4
Key Definitions 5

1.0 INTRODUCTION 6
- **Main Objective** 6

2.0 BACKGROUND 7
- **About ManyPage** 7
  - **What does ManyPage do?** 7
- **A Brief Introduction of the .link file which describes the browsing tree** 8
- **Why WebTree was developed?** 9
- **About the basic functionality of the WebTree** 10

2.0 SOFTWARE REQUIREMENTS 11
- **User Requirements** 11
- **General Description of the requirements** 11
- **Software Requirements Traceability Matrix** 11

3.0 SOFTWARE MODELLING 19
- **Analysis** 19
- **Context Diagram of the Software** 20
- **Domain Chart** 21
- **Information Model** 22
  - **Object Descriptions and Attributes** 24
- **State Models** 27
- **Process Models** 29

4.0 IMPLEMENTATION 32
- **Notes on Design & Implementation.** 32
- **Notes on Implementation "Bugs", Testing and Problem Solving.** 33
- **Future Work** 35

APPENDIX **36**

User's Manual **36**

Acknowledgement

I wish to acknowledge Mr. Pascal Vuylsteker for guiding me through the course of this project and for his ideas, which stimulated much of the present report and the software.

I also gratefully acknowledge the helpful comments of Dr. Henry Gardner on early drafts of this report and the guidance and advice provided throughout this year.

Key Definitions

Browsing Tree - The Browsing Tree refers to the `.link` file structure (or the navigation hierarchy), which is constructed using the local `.link` files.

Directory Tree - The Directory Tree refers to the directory file structure.

`Global.glblink` - There is a unique file associated with this program for each project (i.e. the web site the webmaster is working on). This file is used as a mediator between exporting `.link` data to local files and also for obtaining data about the local `.link` files.

`.link` - This is the file associated with ManyPage which contains link data (i.e. the navigation hierarchy).

local `.link` file refers to the `.link` file which is kept in any given sub directory within the source directory.

Source Directory – This is referred to as the production side, which implies that the directory contains the plain HTML files.

Webmaster – A person who manages a web site.

WebTree - This is the name of the program.

1.0 INTRODUCTION

Main Objective

ManyPage is a Perl tool, which enables webmasters to develop and manage large, multilingual, graphical web sites with rich menus, without the need to use any complex databases.

The main goal of this project is to provide a graphical user interface to create the browsing tree (or navigation hierarchy) needed for ManyPage.

2.0 BACKGROUND

About ManyPage

What does ManyPage do?

ManyPage is associated with creating and maintaining web sites. A website can range from one or two pages to thousands of pages, depending on how much information it provides to the web surfer. It can be cumbersome for a "Webmaster" to manage a large web site particularly when different language versions of the document must be maintained. ManyPage aims to help by reducing the complexity of the management problem and to add more functionality (via rich menus etc) to web pages. In order to use ManyPage the webmaster has to configure the three special files: "`.dress`", "`.obj`", and `.link`. The `.dress` file describes the main template, the `.obj` file describes the objects used in the template and the `.link` file describes the browsing tree. More details will be given below.

To specify a design for a site, we have to define a template (the "`.dress`" file) : a simple HTML file that structures the information, in which we will put HTML objects (like a browsing menu) and data from the source plain file. The "`.dress`" file contains the target web page structure. The "`.obj`" file contains the object definitions used in the template or in source files. These three special files may appear anywhere inside the Source Directory. There should be at least one "`.dress`" and certainly one "`.obj`" file in the root directory.

Please refer to the diagram below, which describes the basic steps that need to be accomplished in order to use ManyPage:

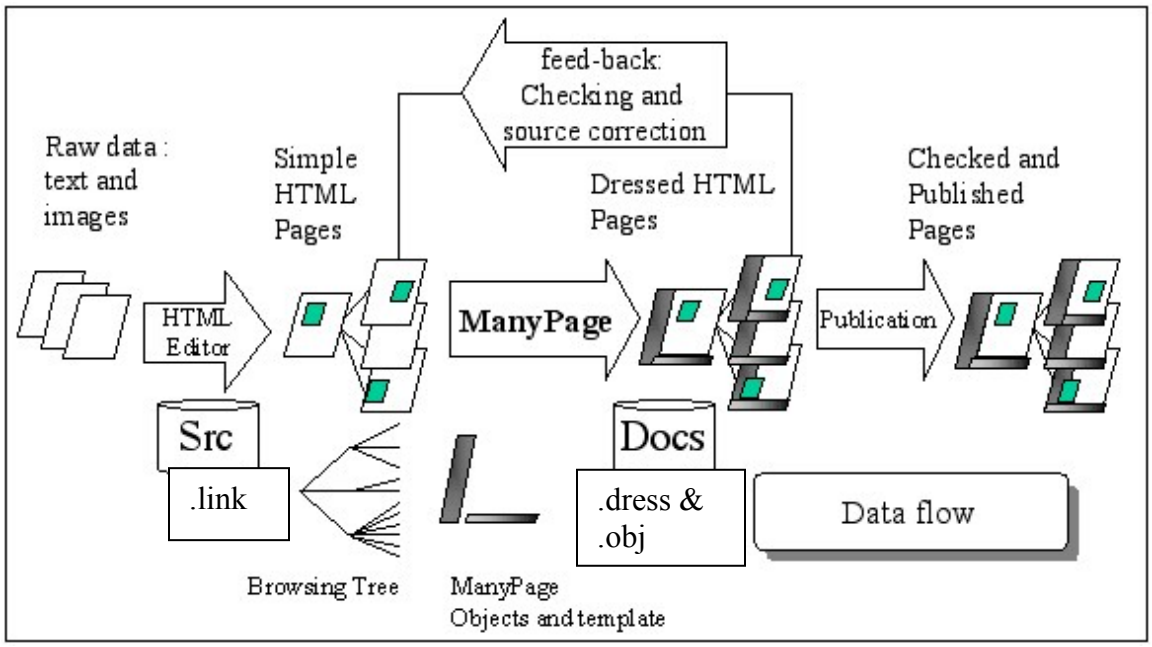


Figure 1

Manypage is used inside a web site production workflow. The webmaster can create the plain HTML pages using an HTML editor (the "source" files). As shown in the diagram the webmaster has to create the three configuration files (`.obj`, `.dress` and the `.link`). After configuring these files the Webmaster can use ManyPage to create the website.

Please Note: For more information about ManyPage refer to
http://www.vrarchitect.net/ManyPage/

A brief introduction of the `.link` file which describes the browsing tree

The `.link` file facilitates dressing the web pages with NEXT/PREVIOUS/UP navigation links.

The Browsing Tree (the contents of the `.link` file) describes the way you want people to understand the hierarchy of your web site. For each web page, you may have one parent and a "NEXT" and "PREVIOUS" page. In a `.link` file, you describe the relative organization of your files within a directory. Below is an example of a visual structure of the `.link` file:

(Note the common extension of the web pages: "`en.html`", means the English version of the html file.)

index.en.html
- : eScience.en.html
- : : eScience.png.en.html
- : teaching/index.en.html
- : research/index.en.html
- : contact.en.html
- : links/index.en.html

Figure 2

For example a person viewing "eScience.png.en.html" will obtain "eScience.en.html" by choosing UP. Then NEXT will move to "teaching/index.en.html" , and "PREVIOUS" will move to "eScience.en.html".

Please note that ':' is used to separate the hierarchy, and between two hierarchies eight spaces are kept (one tabulation). Furthermore, index.en.html is the parent file and eScience.en.html, teaching/index.en.html, research/index.en.html, contact.en.html, and links/index.en.html are the direct sub files (children) of that parent. Also note that the parent of the eScience.png.en.html is eScience.en.html.

`.link` files can be optionally placed in sub directories within the source directory tree to describe the relative organization of the files inside in these subdirectories. This can give rise to problems of inconsistency. Where as it is possible for a webmaster to describe the entire browsing structure of the source directory tree using only one `.link` file in the root node. It is also possible that he or she choose to use several `.link` files to do the same job. For example, if the content of a particular subdirectory was being modified extensively, it would make sense to have a `.link` file in that particular directory which could be updated easily as the content changed. Some of these `.link` files in sub trees may become inconsistent with higher level `.link` files over time.

Please Note: For more information about `.link` refer to
http://www.vrarchitect.net/ManyPage/link.en.html

Why WebTree was developed

The user (typically the webmaster) of the ManyPage tool has to manually edit the `.link` files in order to describe the browsing tree. For example, imagine a large web site containing more than a thousand web pages and directories. When the webmaster tries to use the ManyPage tool for the first time, he or she has to examine every single directory to workout the relative browsing organization of the files in that directory. This can be cumbersome and time consuming. Also, when the user modifies the `.link` files manually, the edited information might be inconsistent with other `.link` files(as described above) or it could contain simple typographical errors.

To overcome the above difficulties, a GUI tool based program is being developed. This tool interacts with the `.link` files so that the user does not have to edit them manually. Providing a graphical tool such as this should reduce the error factor dramatically because it takes care of processing the information about the navigation hierarchy, and eventually, it will ensure that the `.link` files are always consistent with each other.

About the basic functionality of the WebTree

The basic functionality of the WebTree software is that the user can build the `.link` Browsing Tree by selecting a directory, or load the Browsing Tree using the previously saved "`Global.glblink`" file. Then the user is able to Drag and Drop files to modify the navigation hierarchy. After completing the navigation hierarchy, this can be saved to the unique `Global.glblink` file. Then the user is able to export the newly created information to the "local `.link`" files. Afterwards, the ManyPage tool can be used to generate the HTML files with new links.

A diagram is shown below which shows the main window of the WebTree software. To the left, the .links file or the browsing tree or the navigation hierarchy of the web site as given by the `.link` is shown. To the right, the directory explorer is shown, which displays the files and the folders of the source directory. The rest of this report details the requirements, design and implementation of WebTree.

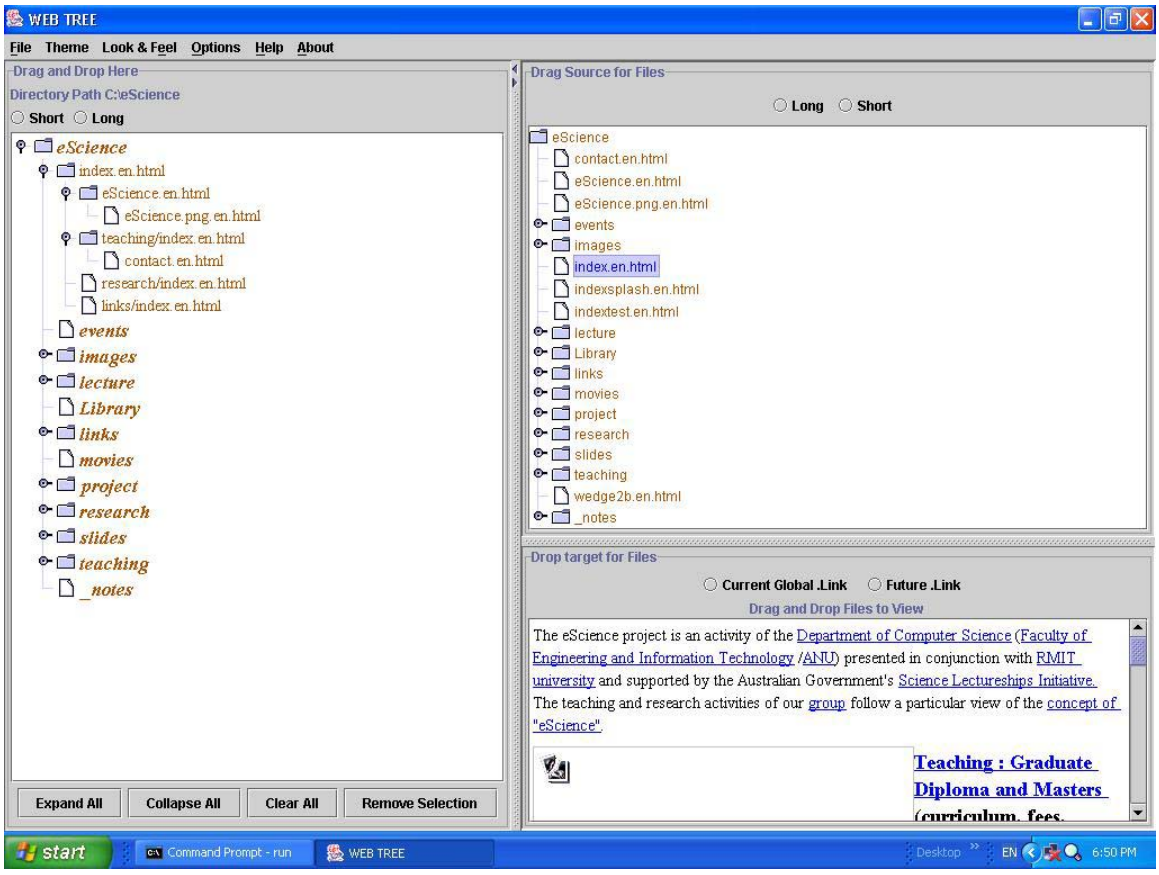


Figure 3

3.0 SOFTWARE REQUIREMENTS

User Requirements

The basic requirements for this program is to create a global hierarchy/navigation (relative to a given directory), to save it in a unique global file (.glblink) and to export that information into the local `.link` files.

General Description of the requirements

Most of the user requirements were derived from the ManyPage tool. These requirements were developed from the documents available on the ManyPage website, (especially the page describing the `.link` file - internet URL www.vrarchitech.net/ManyPage/link.en.html) in combination with discussions with my supervisor. There were no clearly stated requirements in the beginning of this project. It was only a description of the program available, stating the aims of the program. So a great deal of effort was spent developing these requirements.

The requirement of the program was straightforward and it was mainly the ergonomic requirements that remained to be carried out. Hence, making the GUI elements more usable.

Software Requirements Traceability Matrix

Acronyms

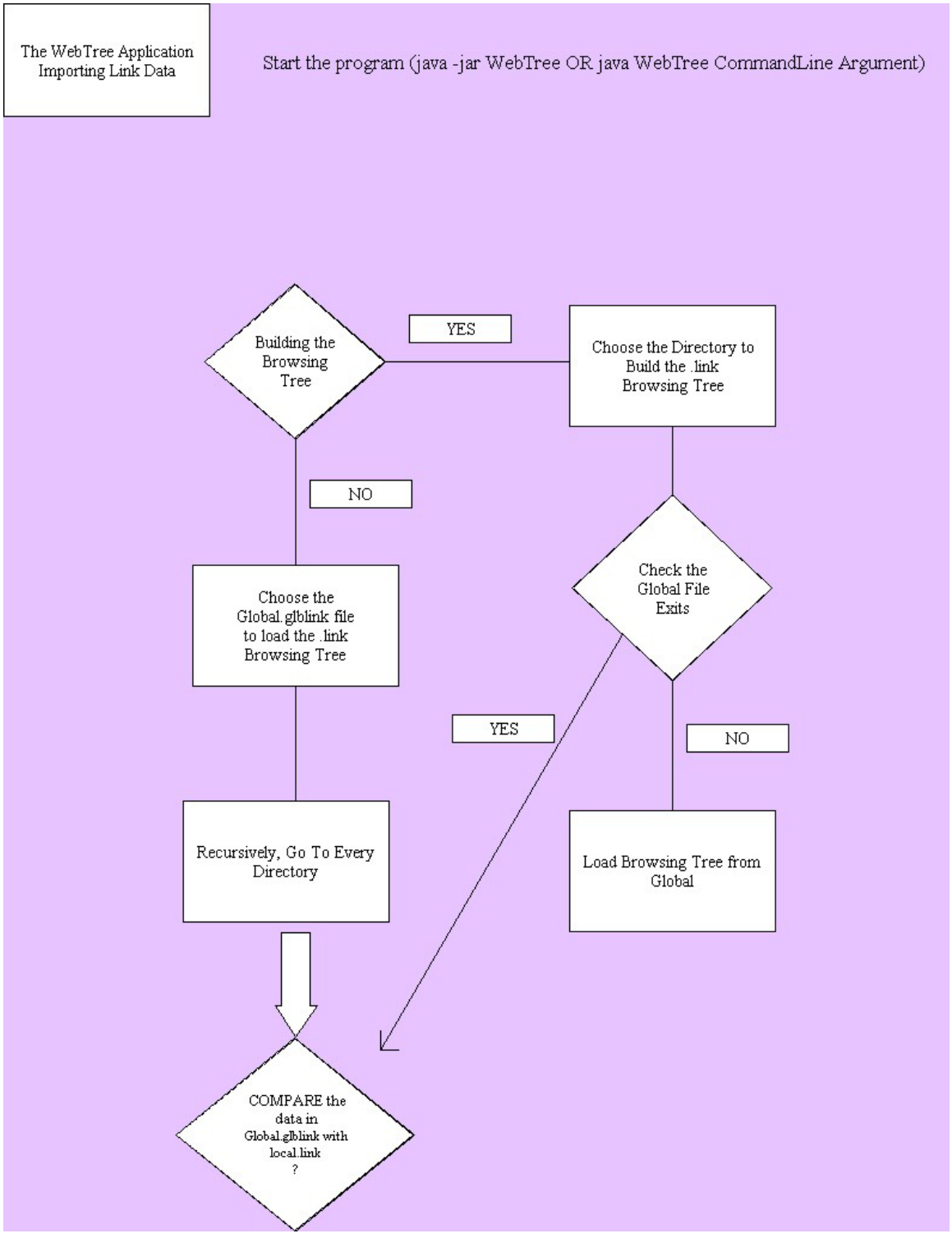
BT - The Browsing Tree
DT - The Directory Tree

Requirement No	Basic Functional Requirement
01	When the application is started the user would have to select the directory that they need to navigate. This means that either the user specifies the directory in which the `.link` data should be loaded, or they can choose the Global `.link` file that is used to obtain the BT.
1.a	The user can execute the program with java executable jar file support. The user can start the program by typing "java WebTree" at the command line or executing the jar file. Type "java -jar WebTree.jar" to execute the jar file.
1.b	The user can input the directory path as the first command line argument or the contents of the current hard disk is shown.

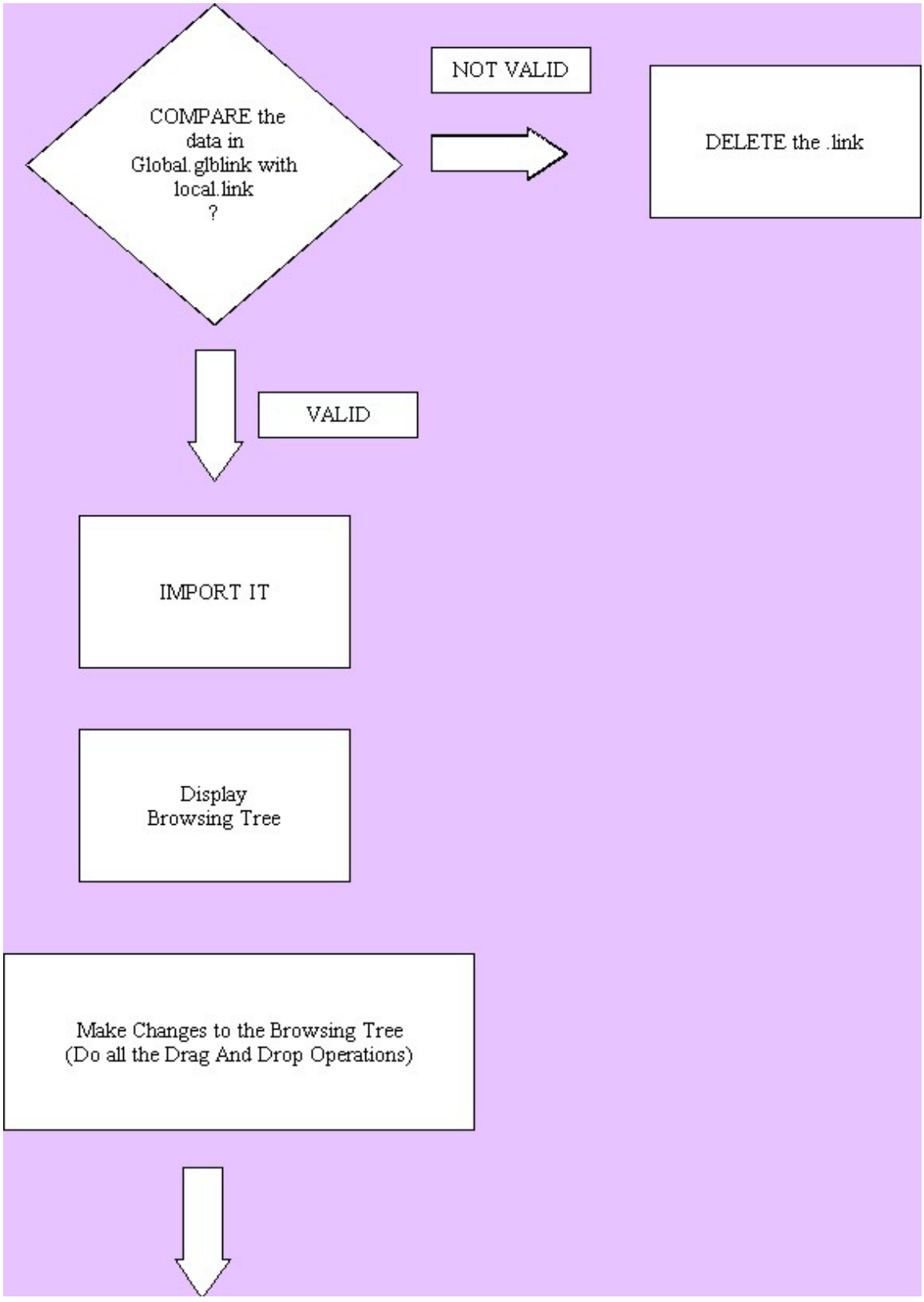
	For example typing "java WebTree ." will result in displaying the current directory in the Directory Explorer. Typing "java WebTree Z:/u3928468/eScience Project " will result in all the files and folders being displayed in the Directory explorer.
2.0	The first step of the program and the other associated steps. Please refer to the flow chart (Figure 4). Note that, this is only an indicative flow chart and it has not been drawn using the proper notation. It is only intended to elaborate the program flow. The flow chart contains three main steps. They are; building the browsing tree, producing the Browsing tree and exporting the Link Data.
3.0	By default the `Global.glblink` file will be saved in the selected directory.
4.0	The changed link data (after the user drag and drop files) will be saved in the `Global.glblink` file. After the user confirms the changes the link data shall be exported to the local `.link` files.
4.a	A configuration file will be asked from the user for each `.link` file that has to be modified.
4.b	A configuration will not be saved for `.link` files that do not require modifying.
4.c	Upon selection to save the configuration, a dialog box will be shown to the user indicating the file being modified. The information about the old `.link` file and the new `.link` file and yes, no and yes to all buttons will be provided in the dialog box.
5.0	There is an "Options" menu and options of the `global.glblink` will be customizable (e.g. where the global should be kept and name of the `Global.glblink` file. etc.).
6.0	As ManyPage allows to manage multilingual websites, there will be an option to select the language. Then a file filter will be used for easy navigation. (For example, if the selected language is French then only the *.fr.html files will be shown).
6.a	The user is able to select potential available languages, but the suffixes .html, .htm will always be shown in the BT. French .fr.html it English .en.html, en.htm

	Spanish .es.html Also the user will be able to add a suffix if needed.
7.0	Back up files will be saved.
8.0	Before modifying the original contents the user must be prompted for incompatibilities.
	Graphical User Interface Description. (Below is a description of the GUI components and their usage)
	The main window consists of two split panes. Below a diagram of the main window is shown (Figure 5). The left side would contain the BT and the right side would contain the DT and a text panel.
	Browsing Tree (BT)
1.0	When the user selects the directory to build the BT, it will be shown in the Browsing Tree navigator. If there is a `Global.glblink` file then it will be displayed in the Browsing Tree also.
2.0	There are GUI control buttons that will be available to help edit the BT quickly. For instance, Expand All - Expands all the nodes of the BT. Collapse All - Shows only the root node of the BT. Clear All - Clears all the nodes of the BT. Remove Selection - Removes the Selected node from the BT. Add Child, Add Sibling.
3.0	Select any node on the BT and right clicking it will result in a popup window being displayed. This popup window would contain Adding a Child Node, Adding a Sibling Node, Removing a Node and all the above GUI buttons described previously.
4.0	Double clicking any node on the BT will result in a Dialog box being displayed. The dialog box is for editing the tags available in ManyPage (TWIN, UP and SEP). There is a ComboBox to select the tag and a text field to input the link information.
5.0	The BT has the drag and drop support (BT is a Drop Target). Any file can be dragged from DT and dropped onto the BT. Also the BT will be a drag source to itself.
6.0	When a file is or has been dragged from the Directory explorer to the

	Browsing Tree, it will still appear with a different colour or shading within the directory tree.
7.0	Selecting a dragged file within the BT will result in it being displayed in the Directory explorer or the BT.
	Directory Tree (DT)
1.0	The Directory Tree will be a Drag Source for Files. Any File can be dragged to the BT or to the Text Panel.
2.0	The contents of the current directory or selected directory will be shown in the directory explorer. The user can set the file types (extension filtering) to be displayed in the DT.
3.0	Selecting a node in the DT will result in the associated `.link` file being displayed in the Text Panel. For instance if the user selects a directory in the DT, and a `.link` file is available in that directory, it will be displayed in the Text Panel.
4.0	Selecting a dragged file within DT will display it in the Directory explorer or the Directory tree.
	Text Panel
1.0	The Text Panel is a Drop Target. Any file can be dragged from the DT onto the Text Panel to view its contents. For instance a HTML file can be viewed on the Text Panel, which has been dragged from the DT.
2.0	The local `.link` file of the currently selected node in the DT will be shown in the Text Panel.


The WebTree Application Importing Link Data
Start the program (java -jar WebTree OR java WebTree CommandLine Argument)
Building the Browsing Tree
YES
Choose the Directory to Build the .link Browsing Tree
NO
Choose the Global.glblink file to load the .link Browsing Tree
Check the Global File Exits
YES
NO
Recursively, Go To Every Directory
Load Browsing Tree from Global
COMPARE the data in Global.glblink with local.link ?

This diagram is continued on the next page:



This diagram is continued on the next page:

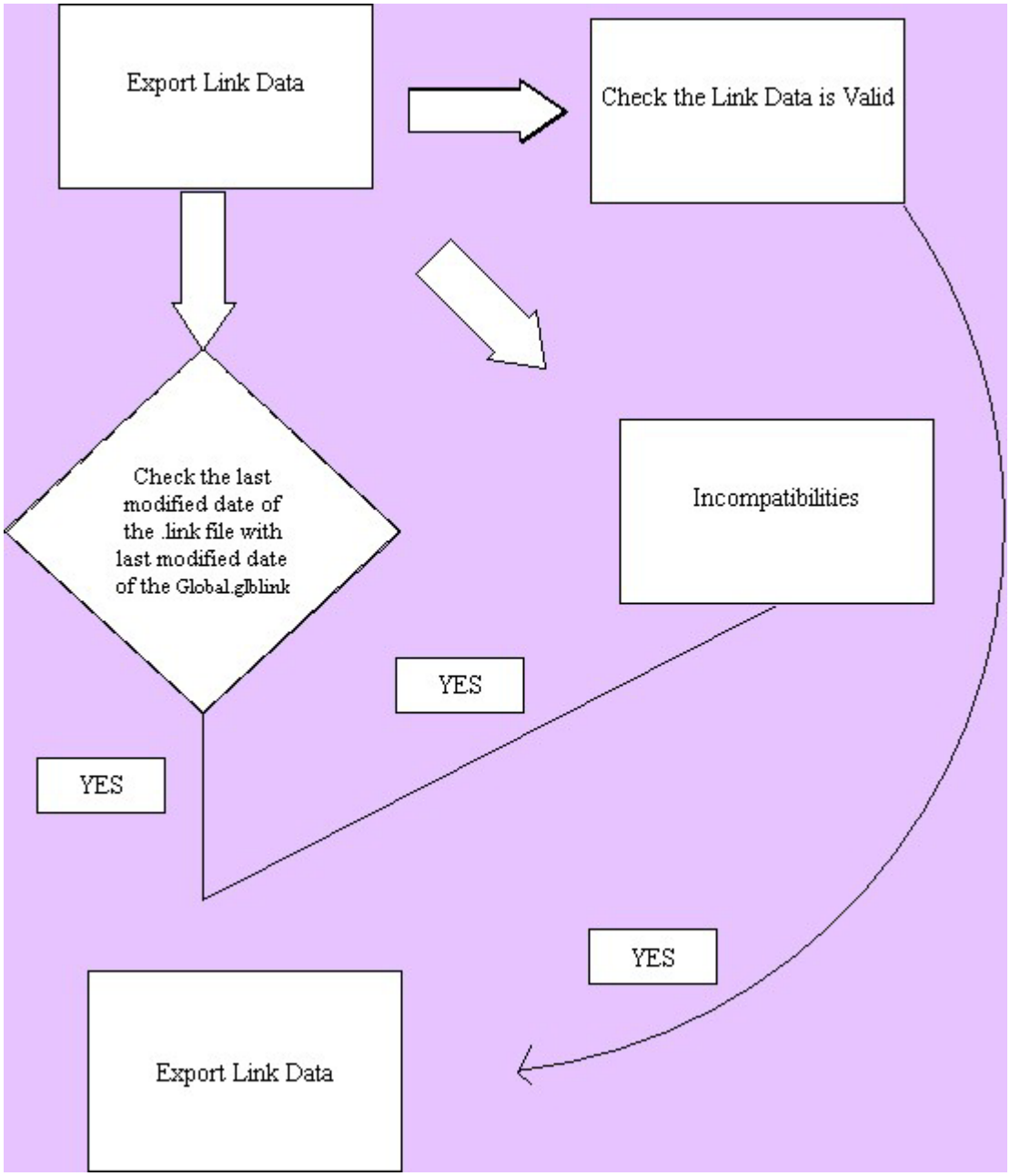

Export Link Data
Check the Link Data is Valid
Check the last modified date of the .link file with last modified date of the Global.glblink
Incompatibilities
YES
YES
YES
Export Link Data

Figure 4

Flow Chart

The layout of the main window of the program.

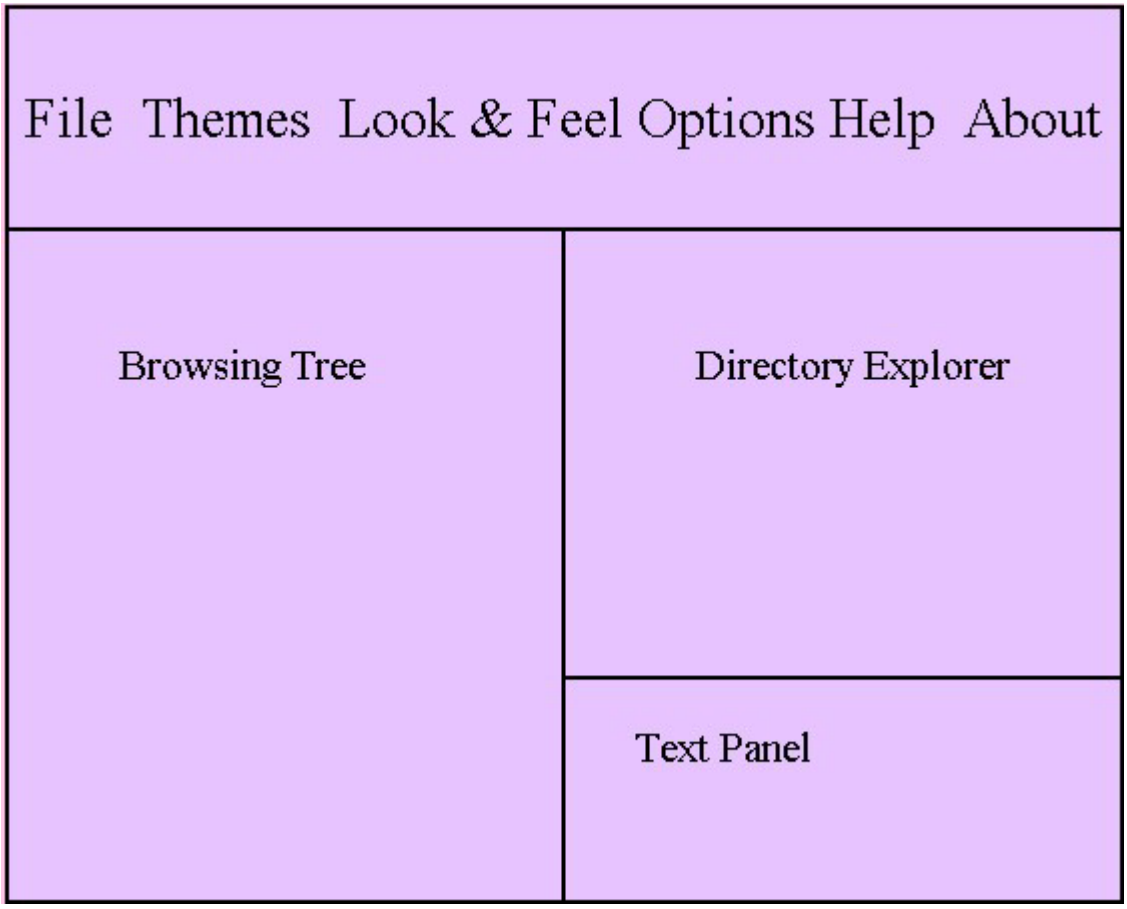


Figure 5

SOFTWARE MODELLING

Analysis

The functionality of the software was mainly described using the Shlaer Mellor method. This method includes how the objects of the software communicate and which states can occur and their important processes.

Shlaer-Mellor Object Oriented Analysis

Using the Shlaer-Mellor method a number of Object Oriented Analysis work products were able to be developed. These include the Domain Chart, Information Model, State Models, Process Models and the Object Communication Model. Please note that these state models and process models are by no means complete. Only some of the main objects state models were modelled and their important process models were derived because these models mainly describe the main functionality of the software.

All these models were drawn using SmartDraw 6.03 version. It has excellent support for drawing Shlaer Mellor models, hence it provides good support for drawing software diagrams.

Context Diagram of the Software

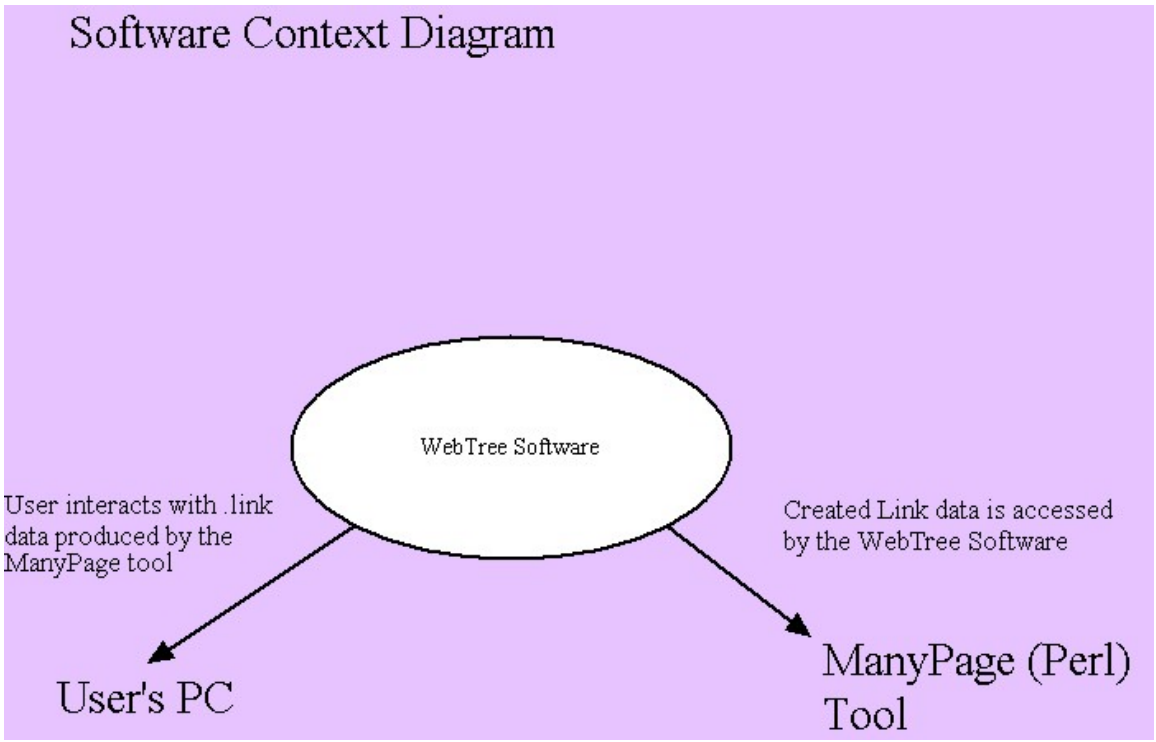

Software Context Diagram
WebTree Software
User interacts with .link data produced by the ManyPage tool
Created Link data is accessed by the WebTree Software
User's PC
ManyPage (Perl) Tool

Figure 6

The software context diagram shows the external entities that the software uses. From this diagram we can clearly identify what kind of interactions it has. As figure 6 shows, the software has two main interactions, one with the user and one with the ManyPage tool.

Some Use cases were developed to describe implementation details. These are by no means complete.

Format of the Use Case is: Use Case ID, who is carrying out the activity, the nature of the action.

UC01 - User - Build Browsing `.link` Tree from Directory
UC02 - User - Build Browsing `.link` Tree Step By Step
UC03 - User - Load Configuration
UC04 - User - Save Configuration
UC05 - User - Load Link Tree from Global File
UC06 - User - Save Changes to Global File
UC07 - User - Export Changes to Link Files

Domain Chart.

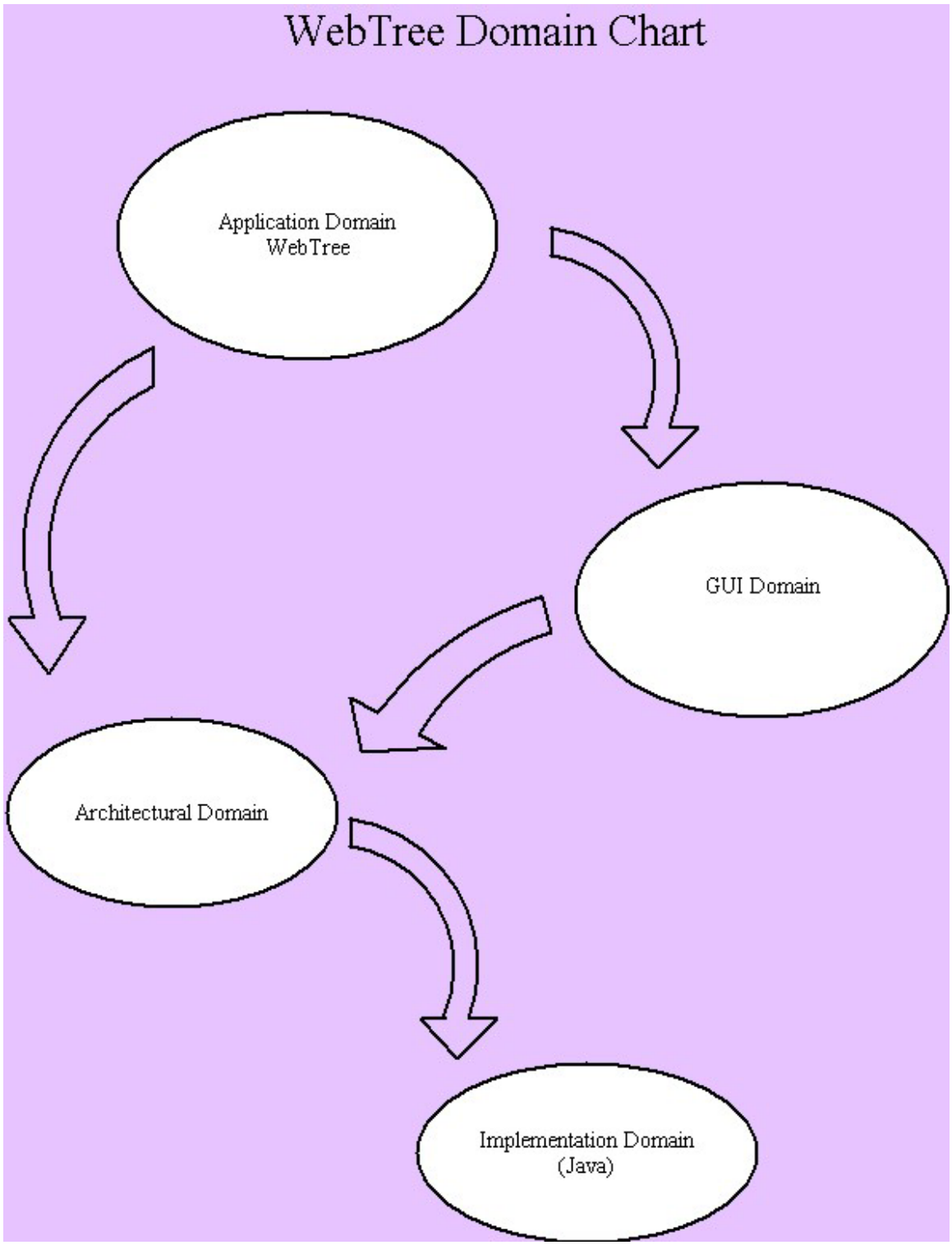


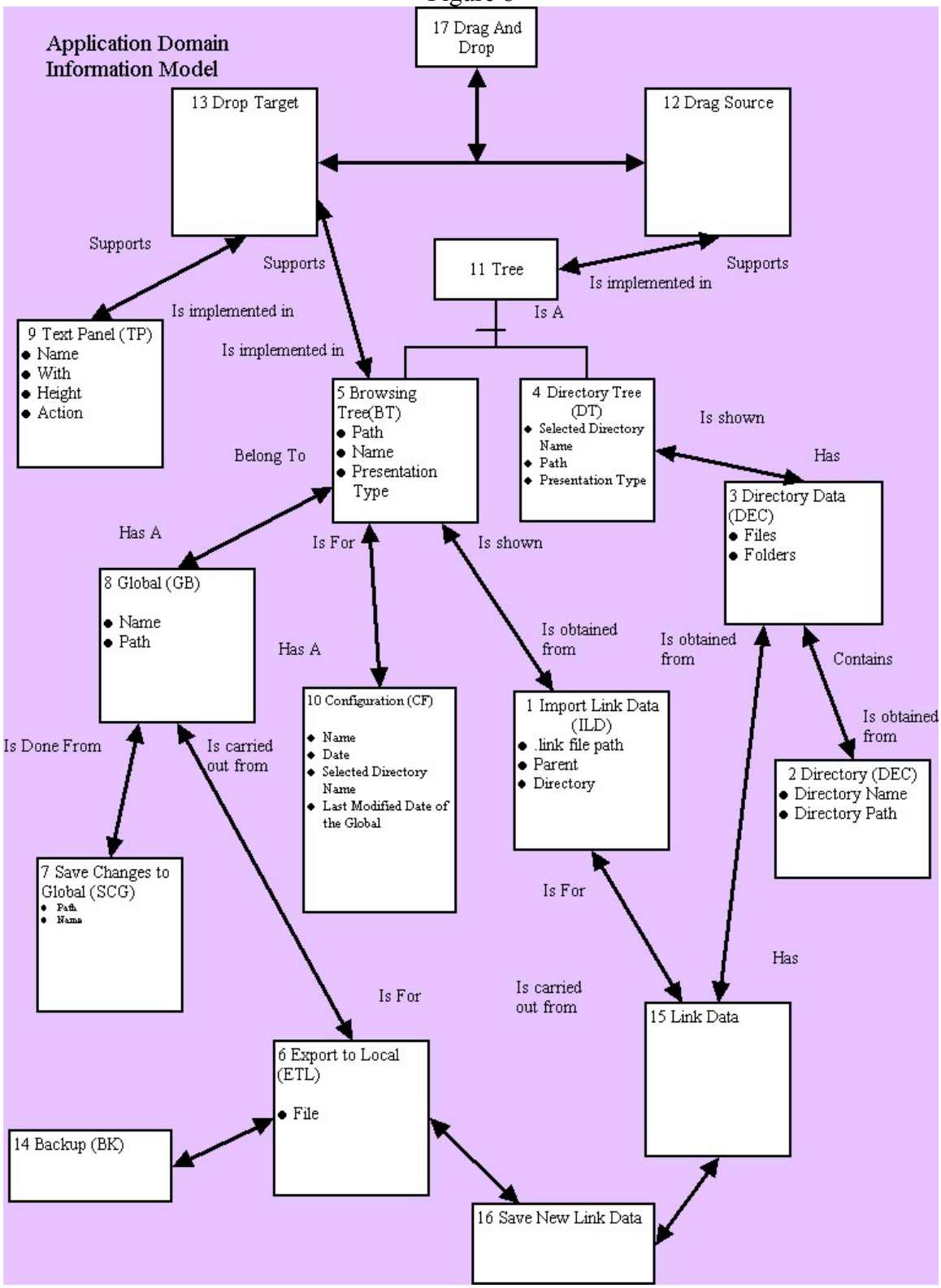
Figure 7

The whole software is broken down to separate sub parts, which describe the main pieces of the software. Using this model it is easier to identify the sub parts, which should be implemented. For instance, GUI domain describes the graphical user interface issues encountered in the software and hence an arrow depicts the use of mechanisms and capabilities provided by another domain.

Information Model.

This model shows the conceptual entities or the objects of the WebTree software. The attributes of these objects are shown inside the box and the relationships between these objects are shown. Identifying the main attributes and the relationship of these objects can be directly used to produce the equivalent java code. For instance, an object having a one to one relationship can be coded by putting the object attributes as the private variables and having a reference to the other object.

Figure 8


Application Domain Information Model
17 Drag And Drop
13 Drop Target
12 Drag Source
Supports
Supports
11 Tree
Is implemented in
Supports
Is implemented in
Is A
9 Text Panel (TP)
Name
With
Height
Action
Is implemented in
5 Browsing Tree(BT)
Path
Name
Presentation Type
4 Directory Tree (DT)
Selected Directory Name
Path
Presentation Type
Is shown
Has
Belong To
3 Directory Data (DEC)
Files
Folders
Has A
Is For
Is shown
8 Global (GB)
Name
Path
Has A
Is obtained from
Is obtained from
Contains
Is obtained from
10 Configuration (CF)
Name
Date
Selected Directory Name
Last Modified Date of the Global
1 Import Link Data (ILD)
.link file path
Parent
Directory
2 Directory (DEC)
Directory Name
Directory Path
Is Done From
Is carried out from
7 Save Changes to Global (SCG)
Path
Name
Is For
Has
Is carried out from
Is For
15 Link Data
6 Export to Local (ETL)
File
14 Backup (BK)
16 Save New Link Data

Object Descriptions and Attributes.

1.Import Link Data (ILD)

Represents the process of importing the browsing tree information.

Directory Path	The system dependant directory path.
Global file path	The of the Global file.

2. Directory (DEC)

Represent the directory, which is selected by the user to obtain the Browsing Tree information.

Directory Name	Name of the selected Directory
Directory Path	System dependant directory path

3. Directory Data (DED)

An object that represents the data obtained from the selected directory. Mainly files and sub folders.

Files	The files in the selected directory
Folders	The sub folders in the selected directory

4.Directory Tree (DT)

Represent the directory explorer of the WebTree.

Selected Directory Name	Name of the selected directory
Path	System dependant directory path
Presentation Type	The presentation of the Browsing Tree (short, long and medium)

5.Browsing Tree (BT)

Represents the Browsing Tree of ManyPage. After obtaining the Link Data, the Browsing Tree is shown.

Path	Browsing Tree path
Name	Name of the Browsing Tree
Presentation Type	The presentation of the Browsing Tree (short, long and medium)

6.Export To Local (ETL)

This represents the process of exporting information to the local `.link` file.

File	Retrieves the path where a `.link` file is kept

7.Save Changes to Global (SCG)

This represents saving Browsing Tree information to the Global file.

Path	System dependant path of the Global file
Name	Name of the Global file

8.Global (GB)

This represents the unique `Global.glblink` file.

Name	Name of the file
Path	System dependant file path
Size	The size of the file

9.Text Panel (TP)

This represents the text panel in the WebTree.

Name of the Text Panel	Name of the Text Panel
Width	Width of the text panel
Height	Height of the text panel
Action	The supported Action (Copy, Paste etc)

10.Configuration (CF)

This represents the configuration created, saved or modified.

Name	Name of the configuration
Date	Date of the creation.
Selected Directory Name	System dependant directory path
Last Modified Date of the Global.glblink	Last Date of the `.link` file information is saved to the Global.glblink

11.Tree

12.Drag Source (DS)

This represents the drop source support.

Name	Name of the drop source

13.Drop Target (DT)

This represents the drop target support.

Name	Name of the drop target

14.Backup (BK)

An object whose role is to create a backup copy of the `.link` before being modified or created. This object has no attributes, it only represents a role.

15.Link Data (LD)

This object represents the contents of the `.link` file, namely, the hierarchical structure of the `.link` file.

`.link` file path	System dependant file path.
Parent	The parent of the hierarchical structure.

State Models.

A State model shows the behaviour of the object. Please note, only some of the objects' state models are shown below even though other objects might have a dynamic behaviour. Hence, the state models of the Import Link Data and Export to Local are shown below. These two objects have the main interest of this software as described in the main functionality of the WebTree.

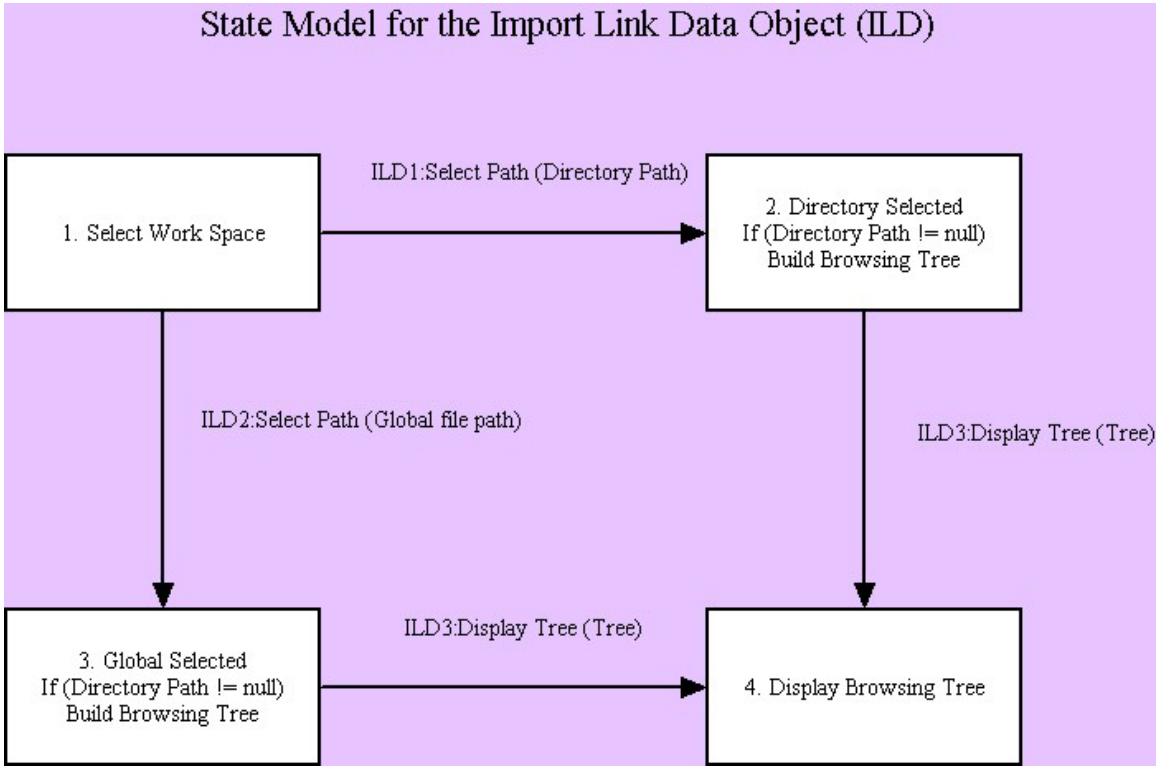

State Model for the Import Link Data Object (ILD)
1. Select Work Space
ILD1:Select Path (Directory Path)
2. Directory Selected
If (Directory Path != null)
Build Browsing Tree
ILD2:Select Path (Global file path)
ILD3:Display Tree (Tree)
3. Global Selected
If (Directory Path != null)
Build Browsing Tree
ILD3:Display Tree (Tree)
4. Display Browsing Tree

Figure 9

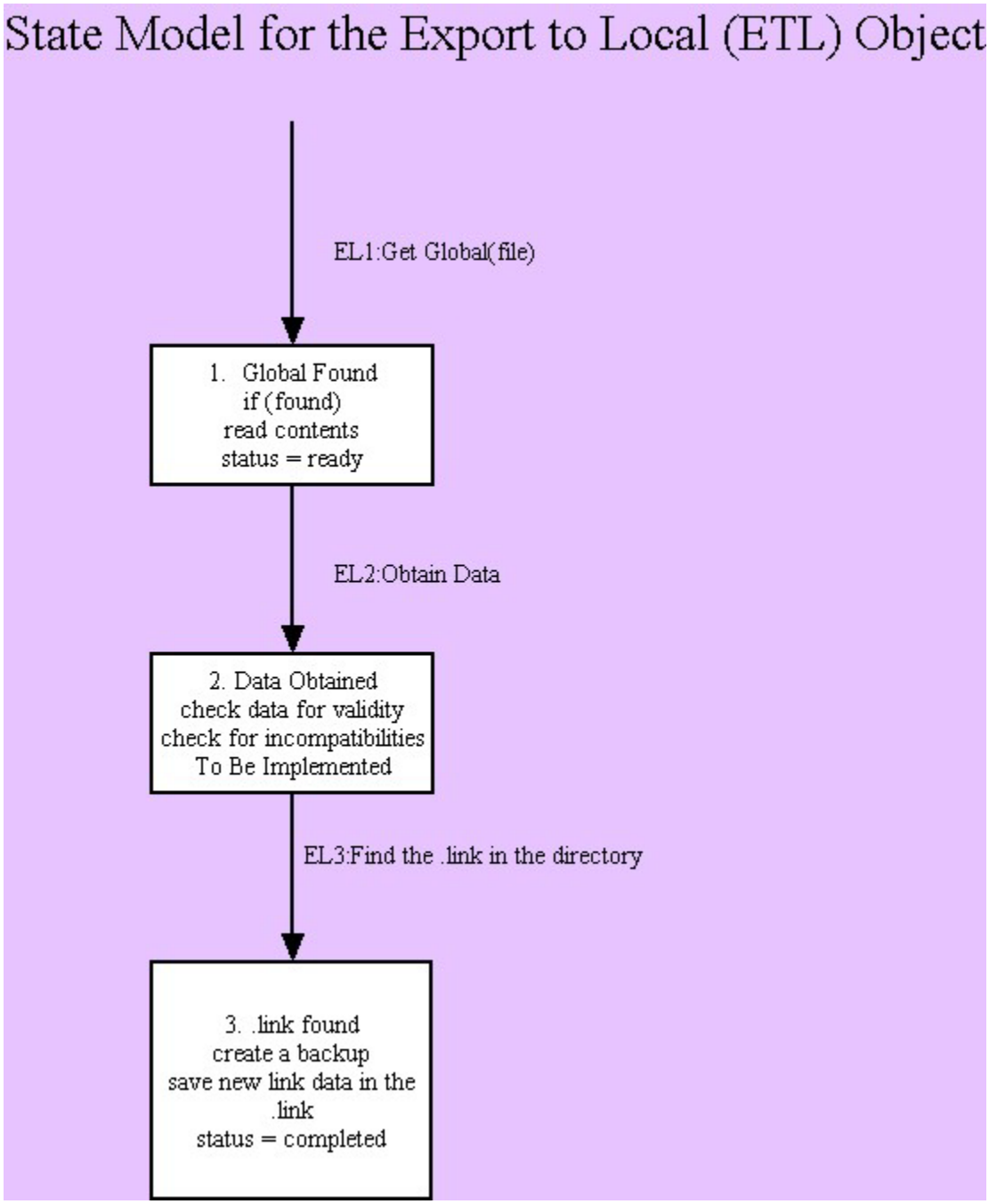

State Model for the Export to Local (ETL) Object
EL1:Get Global(file)
1. Global Found
if (found)
read contents
status = ready
EL2:Obtain Data
2. Data Obtained
check data for validity
check for incompatibilities
To Be Implemented
EL3:Find the .link in the directory
3. .link found
create a backup
save new link data in the
.link
status = completed

Figure 10

Process Models

These process models correspond to the Import Link Data object. Its second and third states processes are shown in these diagrams.

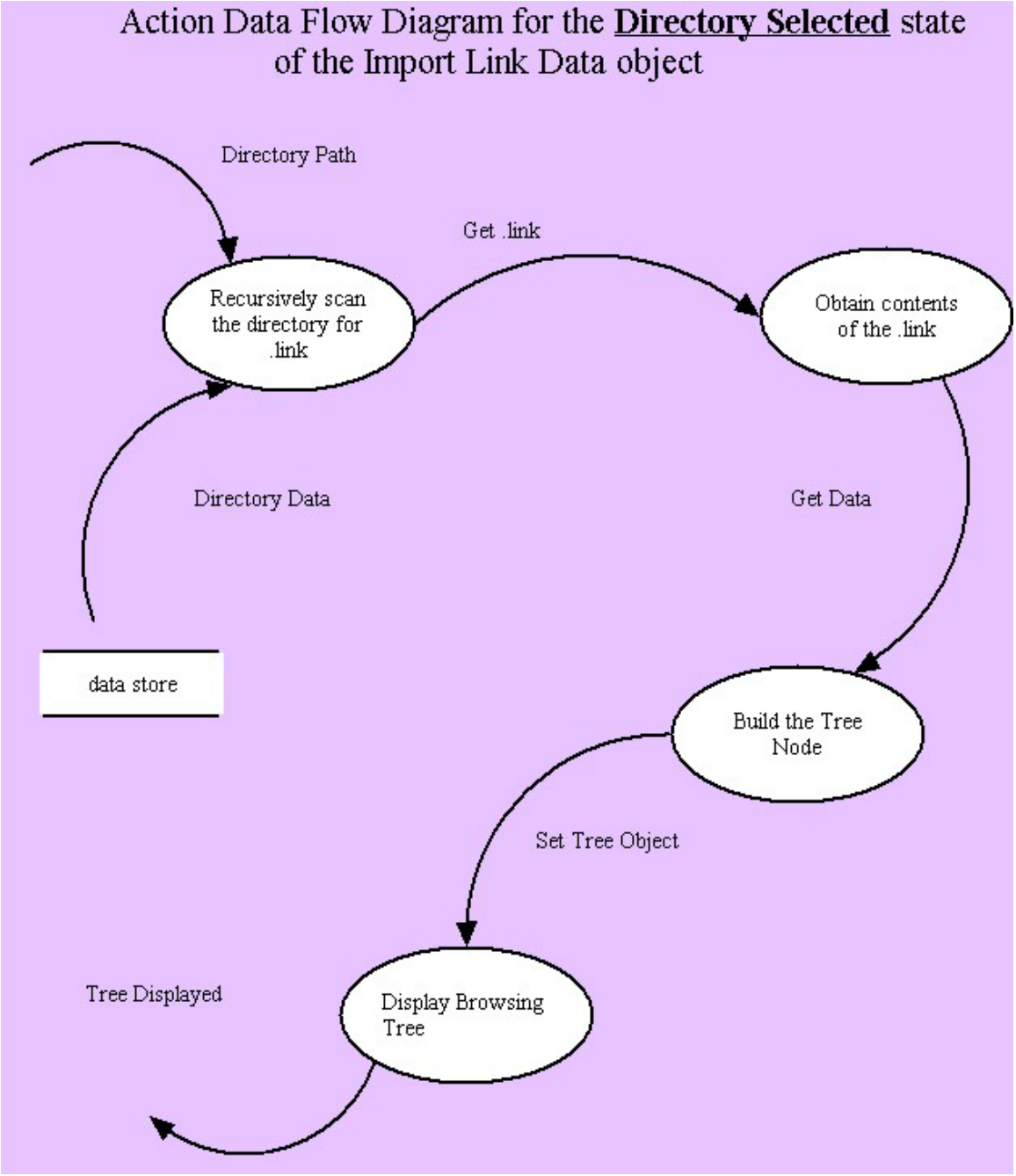

Action Data Flow Diagram for the Directory Selected state of the Import Link Data object
Directory Path
Get .link
Recursively scan the directory for .link
Obtain contents of the .link
Directory Data
Get Data
data store
Build the Tree Node
Set Tree Object
Tree Displayed
Display Browsing Tree

Figure 11

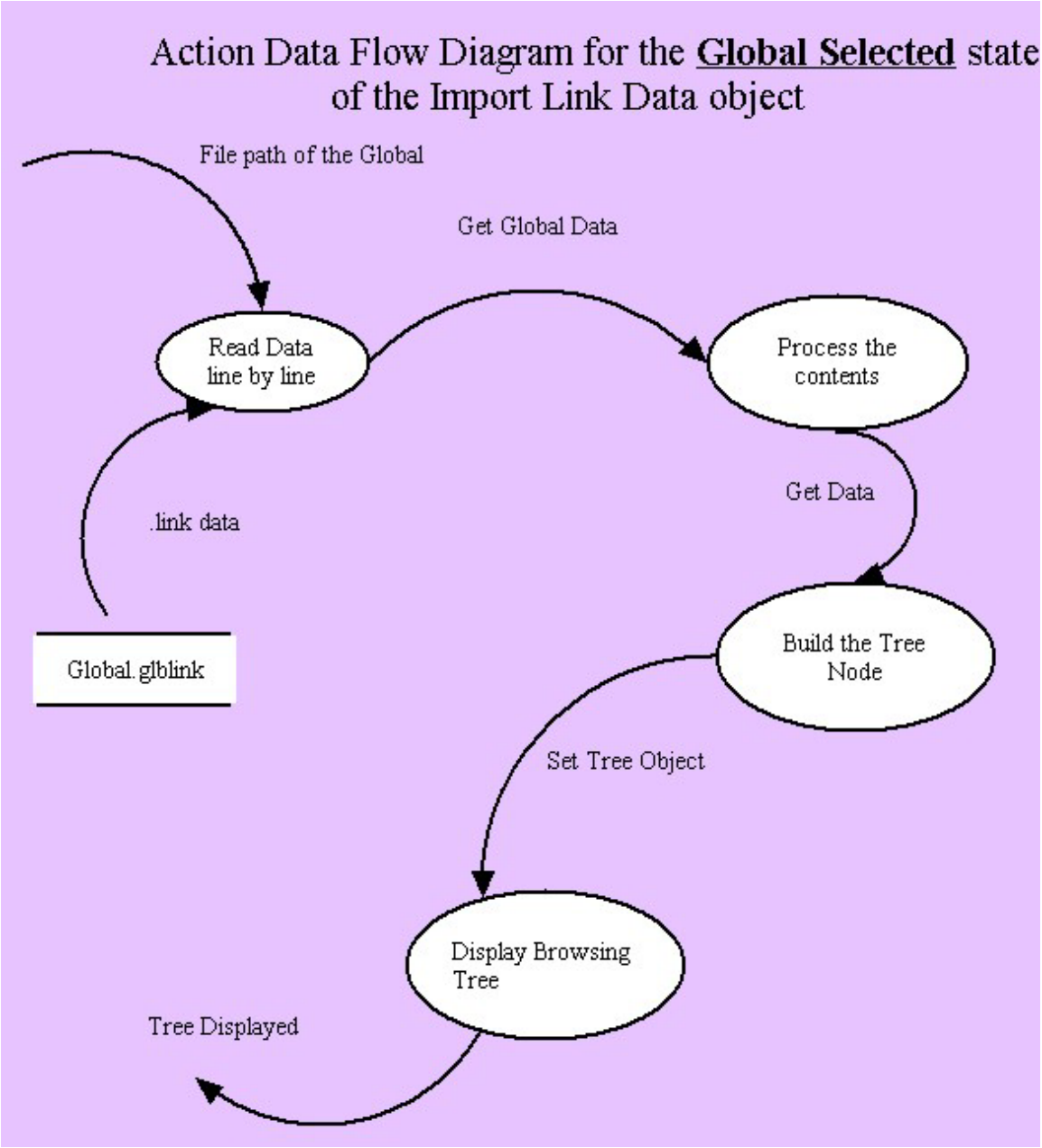

Action Data Flow Diagram for the Global Selected state of the Import Link Data object
File path of the Global
Get Global Data
Read Data line by line
Process the contents
Get Data
.link data
Global.glblink
Build the Tree Node
Set Tree Object
Display Browsing Tree
Tree Displayed

Figure 12

Dialogs

A number of dialogs were required to obtain the user input. The dialog used include:

A file dialog – In one instance, this was used to obtain the directory where the user wants to work. Another was to obtain the Global file where the user wants to load the Browsing Tree.

Configuration – Two dialogs were used to save and create the configuration files.

A dialog to specify the preferences of the program.

A dialog to obtain the tag information and its value.

Java API Swing dialogs like JOptionPane was used for error messages and such.

IMPLEMENTATION

Notes on Design & Implementation.

When I was implementing the program, I used the Software Context Diagram, Domain and the Information model I had created earlier as a guide to design the main classes. However in doing so I discovered some aspects of the implementation details, and hence I had to alter the information model (especially the object relationships). Also the state models of the objects "Import Link Data", and "Export to Local" had to be modified due to issues encountered in their implementation, and their process models also needed to be changed.

After the modelling phase was completed, the implementation of the software was carried out. The prototypes that I had developed during the initial phase of the project with the help from my supervisor helped me immensely. I made the first program prototype during the requirements gathering phase and later showed this prototype to my supervisor for comment. One of the most critical issues of this program was how to make the program usable. The main idea was in making the distinction between recognition rather than recall. I was trying to construct the program, using a direct manipulation style with so that the users can easily recognize how to manipulate the GUI components. This is mainly evident in the utilization of the drag and drop operations and the menu structure. With advice from my supervisor, the drag and drop style of operations and the customized displays of the tree structure were modified.

Creating prototypes of the program is associated with making the GUI components of the program. For that matter Java's Swing Package was used. So the experience I gained from the first semester of the course was used to achieve the above task. Also the experience I gained from Java in the first semester helped me immensely for this project, as Java is not an easy language to grasp quickly. Even though I had used the `JTree` class of the Java API in first semester, the most compelling capabilities of the `JTree` and its other associations were discovered whilst carrying out the project. For example, the drag and drop operation using the `JTree`, using the `DefaultMutableTreeNode` class, using the `DefaultTreeCellRenderer` class, and using the `DefaultCellRenderer` to name a few.

To save the Browsing Tree information to a file (`Global.glblink`), and for the configuration of the program, I thought of using XML. Using XML with `JTree` needs a lot of coding and it needs Java 1.4 edition. However, it is possible to use XML with Java 1.3 by installing a package and adding an appropriate CLASSPATH statement. I found it easier to use a normal file without using XML to save the Browsing Tree information, which is used as the unique `Global.glblink` file in the program.

In the WebTree program, one possible interaction style would be to allow the user to drag files from the directory explorer and to drop onto the browsing tree. This interaction style would enhance the creation of the navigation hierarchy because otherwise, the user would have to look into the source directory and must decide which information should be written in the browsing tree. Presumably, this is somewhat time consuming. So providing drag and drop speeds up the process of creating the navigation hierarchy. If we used another GUI component for this purpose, this task could not be accomplished. For example, using a ComboBox to show the files and folders (Directory Explorer) and allowing it to be a drop source would be somewhat disconcerting to many users.

Notes on Implementation "Bugs", Testing and Problem Solving.

In developing the main algorithm, some of the bugs encountered were due to the poor capabilities of the `StringTokenizer` class. This class has extremely limited capabilities for tokenising. For example, it can be used to tokenize an input with a delimiter but cannot be used to read a hierarchical structure. Hence in most cases, algorithms were developed in conjunction with the use of the `StringTokenizer` class, i.e. the algorithm can read the hierarchical structure and then `StringTokernize` was used to tokenize the input. After developing the algorithms, they were tested over and over again to ensure that they do what they are meant to.

Another annoying bug was encountered using the `String` class. As `String` are immutable objects, when developing the algorithms the `String` variable had the same value. To overcome this, I used the `StringBuffer` class, which is capable of handling mutable strings. Hence the bug was safely dealt with using the `StringBuffer` class.

One of the main features of this program was that most of the algorithms used Java recursion. For instance, this was the case in building and exporting the Browsing Tree structure. Some bugs mainly arose due to runtime errors and incorrect implementation of the recursive algorithm. Therefore, these methods were tested over and over to ensure correct implementation of the recursion. Inside the recursive method, I had to use 'if' statements to filter out the unwanted files.

To test the program, my supervisor sent me to the eScience website, which has been implemented using the ManyPage tool.
I think this was a good source to test the program because it had a lot of `.link` files, so I was able to test the program by checking:

1. Whether it imports the navigation structure of the `.link` kept in any given directory. This means that as the navigation hierarchy is separated using a ‘:’, I was able to test it.
2. Whether it exports the modified navigation hierarchy.

So the whole program was tested using this data. According to the data the program works perfectly well, which implies that this program would be able to produce the correct `.link` file. Also to test a particular implementation, I always used print statements. These print statements involved printing statements such as "System.out.println ("Loop 1 Completed"); " and also printing the values of the objects (For instance "System.out.println (" <TWIN> " + string);"). Also Java's exception handling was used to find some errors. For instance using a try and catch block, I printed the printStackTrace() method to find out the origin of the error. Of course, after the correct implementation of an algorithm, these exception handling and print statements were removed.

In some of the algorithms I used a TreeMap object. The main reason for the use of this class was that it has the ability to store distinct key / value pairs. Hence, it does not duplicate the same value for a given key.

I borrowed the themes (look and feel) used in the Java SDK demo programs, which can be found in the MetalWorks example demo. Using some of these java codes and source files, I integrated the themes to the WebTree program.

When handling the file paths, initially I used the windows separator character '\' to obtain the file. However this program is also used on UNIX and Linux platforms. Therefore, I had to use the File class file separator character (File.separator), which encapsulates the system dependant file separator. This file is implemented as a static String field in the File class.

The most commonly used Java component in this program was the `JTree` class. I extended this class and added more functionality to it. Hence, this required implementation of the Drag and Drop operations with the `JTree`, popup menu, TreeSelectionEvents, Adding and Removing node etc.

When implementing the Text Panel, JEditorPane was used to show the HTML files. JEditorPane has excellent support for HTML/Text file viewing.

Also during the implementation process, these requirements needed to be slightly altered as I realised that they might be used in a more appropriate manner. For instance, my initial thought was to give the user the option to save a `Global.glblink` file with any file name. However, whilst implementing the program it became evident that it would be better to keep the file name unique, as this could reduce errors to a certain extent.

Hints Diary

Please refer to the Hints Diary for more information.

Future Work

The are number of things to be implemented that would enhance the power of the WebTree software. Much of the ergonomic issues have been implemented but there are things to be implemented in order to make WebTree a good intelligent piece of software.

Checking Incompatibilities

When building the browsing tree from the source directory and if a `Global.glblink` file exists, then the information can be checked with the Global file. Depending on how recent the information is the user can be given the option to import the data from the source directory or the Global file.

Exporting Navigation Hierarchy Data

When exporting the information to the local `.link` files, the user can be given the option only to export the modified changes. Hence, a particular `.link` file will not be modified if no information is updated.

Global Hierarchy Navigation

This program only provides exporting information locally, which implies that the user needs to do some thinking when creating the navigation hierarchy. But if a Browsing Tree is built in a Global sense and upon exporting the information an intelligent algorithm can be acquired to export the link data.

Tree Presentations

1. The Short, Medium, and Long representations of the Browsing Tree and the Directory Tree could be implemented.

2. When a file is or has been dragged from the Directory explorer to the Browsing Tree, it will still appear with a different colour or shading within the directory tree.

3. Showing the future `.link` file

Appendix

User's Manual

There are six menus on the menu bar. They are; File, Theme, Look & Feel, Options, Help and About.

The File Menu has eight menu items. They are as follows:

Build Browsing `.link` Tree from Directory
Build Browsing `.link` Tree Step By Step
Load Configuration
Save Configuration
Load Link Tree from Global File
Save Changes to Global File
Export Changes to Link Files
Exit

Build Browsing `.link` Tree from Directory

When the user selects "Build Browsing .link Tree from Directory" a file chooser dialog will be displayed. Select the directory or the source directory from which the user wants to load the browsing tree structure. Hence, selecting a directory will load the structure of the .link files, which are kept in each directory, and all sub directories available in the selected directory.

Build Browsing .link Tree Step By Step

Sometimes when the user selects a directory that contains the data of a website, there could be many files and folders. As a result, a lot of time could be spent on loading the browsing tree structure (also as this method is recursive there is a big overhead). In this case, another loading style is introduced to avoid prolonged delays. Hence, selecting this menu item will only load the first level from the selected directory. Once the first browsing tree level is loaded, the other parts of the tree are loaded according to the tree selection events i.e. if the user clicks a node of the tree, in that instance the recursive method will load the first level of the tree from that point onwards.

Load Configuration

When the user selects to load a configuration file which they had previously saved, it will load the browsing tree without having to do more.

Save Configuration

The user will be able to save a new configuration file which receives the directory being opened and all the parameters will be saved in a properties file.

This menu item can be used to save a configuration file on the work the user would have done so far. Selecting this item will result in a file save dialog being displayed. Please specify a name for the file, select a directory in which the user wants to save the configuration and click the save button to save the file in the location.

Load Link Tree from Global File

If the user does not want to load the browsing tree from a directory, then they can select the unique `Global.glblink` file to load the browsing tree.

Save Changes to Global File

After making changes to the browsing tree, the user can save the browsing tree information to the `Global.glblink` file.

Export Changes to Link Files

The user is forced to save the changes into the Global file first in order to avoid errors. From that point onwards, changes can be exported to the local.link files. Select this menu item after saving the changes to the Global file and then export that newly created browsing tree information to the local`.link` files.

Exit

Quits the program. Please be sure to save the changes, otherwise the changes will not be saved when quitting the application.

Themes menu

Please select any theme you wish!

Look & Feel

Allows selecting the favorite the look and feel.

Options.

Allows selecting the preferences menu item to customize the program.

Help

A help menu shall be used to assist the user. The help menu shall be composed of help topics, a user guide and information about ManyPage. The user can use the guide to learn how to use WebTree software.